International Journal of Information and Computer Science
IJICS Volume 1, Issue 7, October 2012 PP. 188-192
ISSN(online) 2161-5381
ISSN(print) 2161-6450
www.iji-cs.org

Introducing A Simulation Environment For Mobile Agent Middleware Using TOSSIM

Maryam Shoushtari. Master of Science Degree in Software Engineering, Mohammad Hossein Yaghmaee
Dept. of Information Technology and Communication, Payam-e noor University of Tehran, Urmia, Iran
Dept. of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran
shadi_shoshtari60@yahoo.com, yaghmaee@ieee.org

(Abstract) Nowadays, Wireless sensor networks (WSNs(are becoming increasingly attractive to researchers and industries, due to wide range of applications. However, mechanisms to support maintenance, development and execution of applications over WSNs is still remain challenging due to their tight integration with physical world. Middleware can help to bridge the gap between the high-level requirements of the application programs and underlying operations of WSN. Since, mobile agent middlewares such as Agilla has not been transferred to the simulation environment. Thus AgentInjector can be used to inject agents into a network. In this study, we performed a simulation of mobile agents running on Agilla middleware designed for sensor networks. The simulations are performed using TOSSIM assuming that Agilla middleware is installed on the sensor nodes which are running TinyOS operating system. We simulated different agents corresponding to various functions and measured the time it takes for agent software to run in the simulated environment. The results of migration delay and reliability in the simulation of agents is calculated.

Keywords: Wireless Sensor Network; Middleware; Simulation; Mobile Agents; Agilla

1.INTRODUCTION

Wireless sensor networks consist of a lot of sensor nodes, which are low cost, small size and low power supply. WSNs support a wide range of applications. For instance, WSNs can be widely use for environment monitoring, object tracking system, industrial process control and so on. Due to the lack of structure and resource, WSNs also have some limitation such as, resource management, scalability, mobility, security. Middleware solutions are developed as the link between application and low level operating system to solve many wireless sensor network issues [1,2]. There are different middlewares developed for different purposes. Middlewares can be classified considering their programming approaches: database approach, virtual machine approach, adaptive approach and agent based approach [3].

1.1. Database Approach

This kind of middleware approaches treats the whole sensor network as a distributed database. Usually it has an easy interface to use, like SQL that make queries to collect target data. It is good at regularly queries, but lacks to support real time applications, so sometimes it only provides approximate results. Cougar [4], DSWare [5], TinyDB [6] and SINA [6] are members of this category.

1.2. Virtual Machine Approach

Virtual machine middleware approach is used to decrease overall power and resource consumption. The system consists of virtual machines and interpreters. Developers write applications into small modules, and injecting and distributing modules through network, and finally virtual machines interpret the modules to implement application. Maté [6] and Magnet [6] are considered as instances of virtual machine approach.

1.3. Adaptive Approach (Application Driven)

For some specified purpose, application driven middleware could adjusts network configurations according to application requirements. It has a structure that supplies multiple network configurations by choosing suitable protocol in its network protocol stack. Different sensors combinations and network configurations provide different performance of QoS to meet related application requirements. Milan [7] is special purpose middleware which have adaptability.

1.4. Agent Based Approach

The main feature of mobile agent middleware is the applications are treat as modules for injection and distribution through the network using mobile codes. The sensor network can implements tasks by transmitting application modules, and transmitting modules could consumes less power than transmitting big applications. It is efficient approach to support

multi purpose WSNs. Agilla [8] and Smart Messages [9] are considered as instances of agent based approach.

Since mobile agent middleware has not been transferred to the simulation environment, we used a software (Agent Injector) to perform simulation of Agilla middleware on TOSSIM [10] which is a simulation platform of WSN with nodes running TinyOS [11]. These agents, running on the Agilla middleware which is installed on the motes, are allowed to perform different tasks and the performance of the system when running this software is evaluated. The paper is organized as follows: section 2 describes mobile agents and Agilla middleware. In section 3, softwares which are used in this study are presented. Section 4 explains the simulation method and section 5 presents results of the simulations. Conclusions are presented in section 6.

2. MOBILE AGENTS FOR WIRELESS SENSOR NETWORKS

Agents autonomously sense the environment and respond accordingly. Thus agents are very suitable to be deployed in the sensor network environment where many approaches were proposed to solve the well-known problems in sensor networks by using agents. Mobile agents used in WSNs are classical software agents, which are specifically written to run autonomously in adaptive sensor environments. In a WSN, different mobile agents may be constructed for individual tasks, which work cooperatively to accomplish the main objective of the network application [8].

When the agent chooses to migrate to another node in the WSN, it logically transports its state and code to the destination. The mobile agent middleware framework embedded within a sensor network is the fundamental agent-based programming platform of the network. This framework mainly provides the software migration between the sensors. Agilla [8] is the most popular agent-based middleware framework for WSNs. Agilla middleware, can be run as a separate layer above the TinyOS operating system. With the help of Agilla middleware, one or more mobile agents that coordinate with each other are used in the network to achieve a specific goal. Agilla helps on simplifying the deployment of adaptive WSN applications in order to make implementations. Agilla, through one instruction, can realize the agents' movement between sensor nodes by carrying only the code, which is called as weak migration, or by cloning the code and the state together, which is called as strong migration. Agilla maintains local tuple space and neighbor list for sensor nodes to coordinate agent communication. Neighbor list consists of the addresses of one hop neighbors and the tuple space stores data to be shared by the local and remote agents (agents which reside on other nodes) [12] .

3. DESCRIPTION OF SOFTWARES

Wireless Sensor Networks have limited hardware capabilities from the point of power, processor and memory capacities. A WSN operating system must satisfy application requirements. Moreover, it must allow multiple applications to simultaneously use system resources. Existing operating systems do not meet these requirements. Therefore, there was a necessity to develop a WSN–specific operating system. Such a WSN operating system was built at the UC Berkeley, and named TinyOS. TinyOS is an open source and modular operating system that is very well-suited for WSNs.

TOSSIM [13] is a scalable simulation framework for WSNs that run on TinyOS. TOSSIM is part of the TinyOS operating system. Users can test a real TinyOS application by compiling the application into the TOSSIM instead of real motes.

TinyViz [14] is a visualization, actuation and control tool written in Java for TOSSIM. TinyViz can capture all of the events in a simulation and visualize sensor readings, LEDs and radio links. A user can use TinyViz to simulate an application with "tinyviz -run build/pc/main.exe number_of_nodes" command in the application folder.

Agilla's architecture has three layers: TinyOS, Agilla middleware, and mobile agents. TinyOS is the lower layer of the architecture which Geographic routing, network stack and sensor components are managed by that. Middle layer of the architecture is the core component of the Agilla middleware. Five managers together with the Agilla engine compose the middleware. Each of these is implemented as a TinyOS component and a separate process. These managers are tuple space manager, reaction manager, agent manager, context manager and code manager. Mobile agents constitute the top layer of the architecture. A mobile agent is composed of a stack, heap, various registers and code [15].

4. SIMULATION METHOD

In this study, we used TOSSIM to simulate a WSN with agents loaded on each mote. The motes have TinyOS running on them. We have also used TinyViz, a visualization tool for TOSSIM simulator.

To setup the simulation environment, some software must be installed. These are: Sun's Java Development Kit (JDK) version 1.4.x, Sun's javax.comm package. Later, TinyOS, Agilla, TinyViz and AgentInjector can be installed. mobile agent middleware such as Agilla has not been transferred to the simulation environment, Thus Agent Injector can be used to inject agents into a network. First goal in conducting the simulations was to measure performance of important Agilla instructions. The simulations are performed on a 25-node network arranged in a 5*5 grid. The term "migration" consists of moving and cloning of an agent. Moving of an agent is the transferring of the agent from one node to another node, whereas cloning is copying an agent from one node to another node. Special instructions are used to move or clone one agent from one node to another. Cloning instructions are sclone and wclone. Moving instructions are smove and wmove. In order to benchmark strong and weak migrations, we used test agents. We measured consumed time during the smove (strong move), wmove (weak move), sclone (strong clone) and wclone (weak clone) operations.

5. SIMULATION

We used two different scenarios to benchmark move (smove and wmove) and clone (sclone and wclone) instructions. The difference between the scenarios is the heap operations; 10 variables are recorded to the heap in the first scenario while heap operations were not used in the second scenario. The heap is a random-access storage area that can store up to 12 variables. First, we test smove and wmove instructions for each scenario (with and without heap operations). Then, sclone and wclone instructions are simulated for same scenarios.

5.1. smove vs. wmove (with heap operations)

The agent used to simulate smove instruction with heap operations is shown in Figure 1. This agent saves 10 values to its heap and moves to a random neighbor carrying its code, program counter, stack and heap. We simulated these agents on a virtual wireless sensor network with 25 nodes. We repeated the simulations 50 times for each agent (smove and wmove). the average elapsed times for smove and wmove operations were computed. The results are displayed in Figure 1.

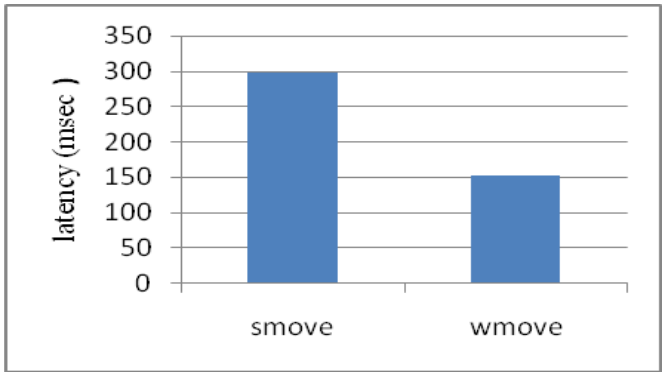

Figure1. Latency of smove and wmove instructions with heap operations

According to Figure 1, average migration time of an agent from one node to another is 298 milliseconds with the smove instruction; while wmove takes 153 milliseconds. This result is expected, because smove instruction transfers agent's heap together with the agent's code, while wmove instruction transfers only the agent's code. Each variable in the heap is 40 bits; therefore, 10 variables use 400 bits. This extra-load increases the migration time of smove agent.

5.2. smove vs. wmove (without heap operations)

In these agents, heap operations are removed to test the amount of the time taken for smove and wmove. We also repeated the simulations for 50 times for each agent (smove or wmove). We then computed the average latency for smove and wmove operations which are shown in Figure 2.

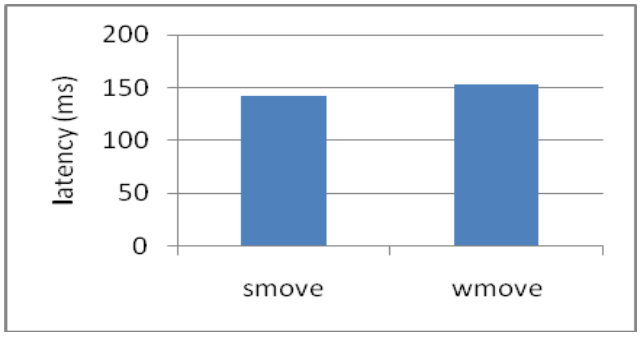

Figure 2. Latency of smove and wmove instructions without heap operations

Average migration time of an agent from one node to another is 139 milliseconds with the smove instruction; while wmove takes 153 milliseconds. This result is expected, because in a weak migration, agent resumes execution from the beginning.

5.3. sclone vs. wclone (with heap operations)

We repeated the simulations for 50 times on a simulated wireless sensor network with 25 nodes as in the previous simulations. Figure 3 displays the results of the simulations for sclone and wclone agents with heap operations.

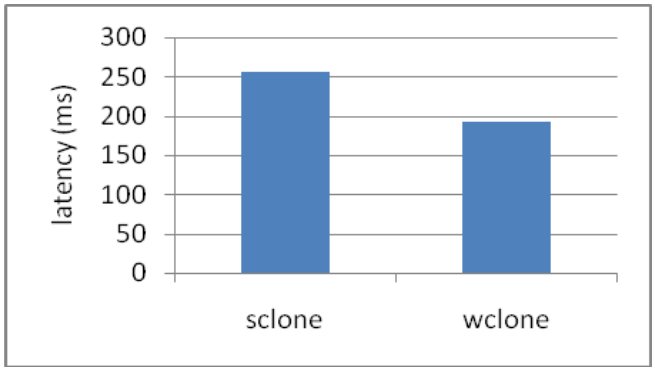

Figure 3. Latency of sclone and wclone instructions with heap operations

According to Figure 3, average migration time of an agent from one node to another is 256 milliseconds with sclone instruction; while wclone takes 193 milliseconds. This result is expected, because transferring agent's code together with the agent's heap increases latency.

5.4. sclone vs. wclone (without heap operations)

According to Figure 4, average migration time of an agent from one node to another is 170 milliseconds with the smove instruction; while wmove takes 193 milliseconds. This result is expected, because in a weak migration, agent resumes execution from the beginning.

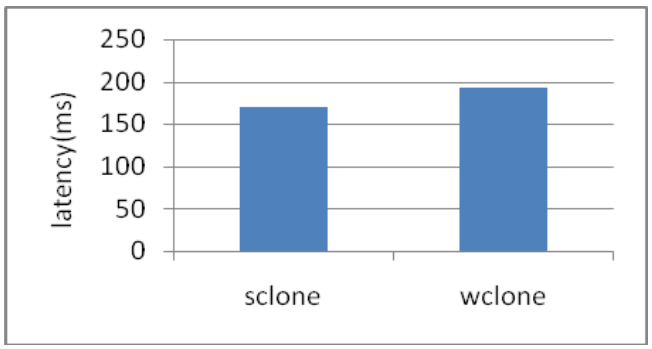

Figure 4. Latency of sclone and wclone instructions without heap operations

6. SPREADING OF AGENT BENCHMARK

To measure the spreading performance of the agents, we created a scenario according to which an agent is injected to a node of the WSN to sense the temperature of the node and send the recorded value to the base station, and the agent clones itself to its neighbors for spreading into the network. Neighbor nodes are chosen randomly by *randnbr* instruction. This instruction looks at the neighbor list of the agent and chooses a random location from the list. The goal of the agent is to visit all the nodes of the WSN and send temperature readings to the base station. Clone operations were preferred to provide faster spreading. Since *wclone* instruction is more efficient than *sclone* as mentioned before, it was chosen for migration.

This agent has been tested on simulated wireless sensor networks which included various number of nodes. The agent was tested 10 times on each WSN. Average values of the tests are taken as the final result. Simulation results are shown in Figure 5. The graph shows that the agent can spread into a 5-node Wsn in 0.4 seconds while it can spread into a 15-node WSN in 1.3 seconds. As expected, larger WSN increases spreading time. On the other hand, latency per node is decreased as the network size increases. For example, 115 ms is consumed per node for a 5-node WSN, while 80 ms is consumed per node for a 150-node WSN.

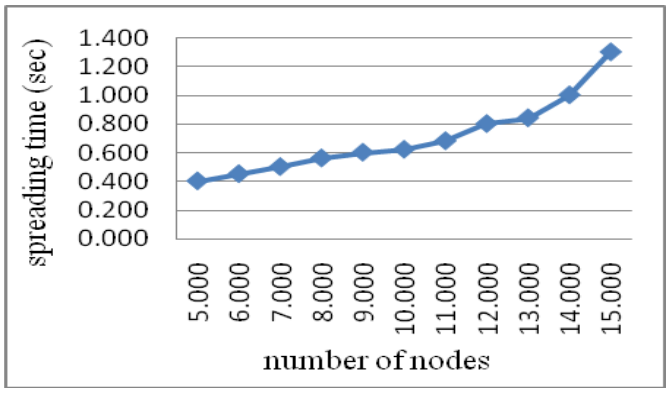

Figure 5: The Agent's Spreading Time into WSN

As mentioned before, the agent clones itself to its neighbors randomly. Accordingly, the agent is cloned to some nodes more than once while other nodes aren't visited by the agent. Reliability of this agent is computed according to the equation:

$$\text{Reliability} = \frac{\textit{Total Number of Nodes} - \textit{Number of nonvisited Node}}{\textit{Total Number of Nodes}} \quad (1)$$

Result of the reliability analysis is shown in Figure 6. Reliability of the agent changes between 76% and 85% inconsistently because of the random nature of the migration operation.

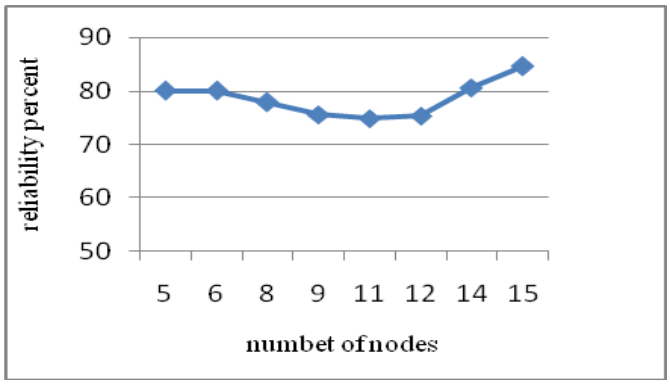

Figure 6: Reliability of the agent in Figure 5.

7. CONCLUSIONS

This study concerned with the lack of simulation environment for mobile agent middlewaes in TinyOS and presented a simulation environment for mobile agents running on Agilla using TOSSIM. We tested different Agilla commands and measured the execution time of each agent using simulation. By utilizing TinyViz to visually observe the simulation process and AgentInjector to transfer Agilla to the simulation environment, we tested different Agilla commands and measured execution time of each agent using simulation and the results were analyzed for simulation validation process.

These experiments will be carried out using different network topologies and different agents in the future. Actual network will be setup and experiments will be compared with simulations as future work.

REFERENCES

[1] Wang, M., Cao, J., Li,J., and Sajal K. Dasi. (2006)." Middleware for Wireless Sensor Networks: A Survey" .Department of Computer Science, Internets and Mobile Computing Lab, Department of Computer Science and Engineering,, Supported by Hong Kong Polytechnic University under the ICRG grant NO.G-YE57, Hong Kong RGC under the Grant of A Research Center Ubiquitous Computing and the National Hi-Tech Research and Development 863 Program of China under Grant No.2006AA01Z231.

[2] Md.Atiqur.R.(2009)."Middleware for wireless sensor networks Challenges and Approaches". TKK T-110.5190 Seminar on Internetworking.

[3] Tong, S. (2009)." An Evaluation Framework for middleware approaches on Wireless Sensor Networks " . Seminar on Internetworking. Helsinki University of Technology, TKK T-110.5190.

[4] MC Lin, YC Chen, SL Tsao. (2012). " Design and implementation of a home and building gateway with integration of nonintrusive load monitoring meters " Industrial Technology (ICIT), 2012 IEEE International Conference. 148-153.

[5] Prasannasrinivasa.S and Suman.N. (2012). "A Study of Middleware for Wireless Sensor Networks". International Journal of Research and Reviews in Ad Hoc Networks (IJRRAN) Vol. 2, No. 2, June 2012, ISSN: 2046-5106.

[6] Kumar.A, Xie.B. (2012)." Handbook of Mobile Systems Applications and Services. "University of Louisville,Louisville,KY,USA , University of Cincinnati, Cincinnati, OH, USA .

[7] Md.Atiqur.R. (2009). "Middleware for wireless sensor networks Challenges and Approaches". Helsinki University of Technology, Seminar on Internetworking. TKK T-110.5190.

[8] Fok, C.-L. , Roman, G.-C. , and Lu, C. (2009). " Agilla: A mobile agent middleware for self-adaptive wireless sensor networks " .ACM Trans. Autonom. Adapt. Syst. 4, 3, Article 16 (July 2009), 26 pages.DOI = 10.1145/1552297.1552299

[9] Kang, P., Borcea, C., Xu, G., Saxena, A., Kremer, U., & Iftode, L. (2004). "Smart messages: A distributed computing platform for networks of embedded systems " .*The Computer Journal, 47*(4), 475-494. doi: 10.1093/comjnl/47.4.475.

[10] Levis, P., Lee, N., Welsh, M., & Culler, D. (2003) ".TOSSIM: accurate and scalable simulation of entire TinyOS applications”**.** *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, 126-137. doi: 10.1145/958491.958506 **.**

[11] Levis, P., Madden, S., Polastre, J., Szewczyk, R., Whitehouse, K., Woo, A., et al. (2005) . " TinyOS: an operating system for wireless sensor network" . In J. R. W. Weber, and E. Aarts (Ed.), *Ambient Intelligence*. New York, NY, SA: Springer-Verlag.

[12] Dagdeviren.O. , Korkmaz.I., Tekbacak. F., Erciyes.K... (2010)."A Survey of Agent Technologies for Wireless Sensor Networks”**.** Izmir Institute of Technology, Computer Eng. Dept., Urla, Izmir 35430, Turkey. Izmir University of Economics, Computer. Izmir University, Computer Eng. Dept., Uckuyular, Izmir, 35350, Turkey.

[13] Levis, P., & Lee, N. (2003). " TOSSIM: a simulator for TinyOS networks" Berkeley, CA, USA: Computer Science Division, University of California Berkeley**.**

[14] Zia, T., & Zomaya, A. (2005). "An analysis of programming and simulations in sensor networks " . *Proceedings of the 1st International Workshop on Sensor Networks and Applications*, 15-19.

[15] Fok.C-L. , Roman.G-C. , and Lu.C. (2005). “Rapid Development and Flexible Deployment of Adaptive Wireless Sensor Network Applications ".Washington University in St. Louis Saint Louis, Missouri 63130.